On some strategies for missing values in positive semidefinite matrices

Lluís A. Belanche
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
c/ Jordi Girona, 1-3
08034 Barcelona
belanche@lsi.upc.edu

Miguel Vázquez
Dept. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
c/ Prof. José García Santesmases, s/n
28040 Madrid
mivazque@fdi.ucm.es

Abstract

This article presents our work on missing values in Positive Semi-Definite or PSD matrices. We show how simple properties of PSD matrices can be used to deal with missing values. We study several situations and investigate their applicability to support vector machine (SVM) problems. In order to illustrate the methods, a set of experiments is presented and discussed.

1 Introduction

A Support Vector Machine (SVM) [1] takes a set of input data examples and builds a model by means of a certain optimization algorithm and the use of a Positive Semi Definite (PSD) kernel function. This function acts on the inputs and computes a real number that accounts for some measurement of symmetric similarity between a pair of examples. Requiring the kernel function to be PSD is very often a mathematical condition for the optimization algorithm [2]. The SVM can then be understood as implicitly building a square symmetric matrix K from the data, where each matrix entry k_{ij} is the value computed by the kernel function for the input examples i and j. Matrices computed in this way are usually called *kernel matrices*.

This work focuses in the issue of *kernel matrices data corruption*. Corruption may appear in several forms and have different causes. A possible categorization is as follows:

Type I A *complete* matrix K_1 that is not PSD, but is the result of a process that *in theory* should yield a PSD matrix K_0. This may be due to numerical precision errors, storage or transmission errors, or simply because the original data examples are incomplete. In this latter case, only the common complete values are used to generate each entry for K_1 (that is, if there were no missing values, then $K_0 = K_1$). This occurs very commonly when estimating covariance matrices out of incomplete data. Of course, these scenarios are not mutually exclusive.

The goal is to turn K_1 into a PSD matrix K_2 that is closest (in some precise sense) to K_0.

Type II A *complete* matrix K_1 that is not PSD, because is the result of a process that *should not* yield a PSD matrix (but a nearly PSD one). This may happen whenever attempting to use a kernel that is "almost always" or "nearly" PSD.

The goal is to turn K_1 into a PSD matrix K_2 that is closest (in some precise sense) to K_1.

Type III An *incomplete* or partial matrix K_1, arising whenever we are directly given a pre-computed kernel matrix with some entries missing. It is usually the case

Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005, pp.159-168
ISBN: 84-9732-449-8

when the entries are the result of a measuring or judgement process that itself yields similarity values (as in certain cognitive tasks [3]). It often implies no knowledge of the kernel function with which to generate the matrix.

The goal in this case is to build K_2 as a complete version of K_1 that is both PSD and closest (in some precise sense) to K_1.

The problem often encountered in practice is of type I and is most likely caused by missing values directly in the input data. In this case standard data imputation techniques can be used (e.g. list or pair-wise deletion, mean substitution, simple regression or the expectation maximization algorithm, among others). These imputation techniques are very likely to yield non-PSD matrices.

In this work, we propose a method to fill in the missing values in kernel matrices that makes use of an observed property of PSD matrices, and as such it is directly applicable to type III situations. The new method is presented in section 3, and turns out to be our main contribution. Actually the problem is reduced to a type II situation. It can then be shown by standard arguments (section 2) that the obtained complete matrix can be made PSD so that it resembles the most the original one (as measured by the Frobenius distance among matrices). Finally, in section 4, we present some experimental work that covers the different situations and tries to illustrate the approaches.

It should be mentioned that our research is oriented toward SVM classification. However, we hope that the same principles can be applied to other applications in which PSD matrices are used. To this end, in some points along the discussion we make some additional considerations that could be interesting in other contexts. We also provide with pseudocode –for the most relevant calculations– in the Appendix.

2 Forcing the PSD property

Let us consider a process that is expected to produce a PSD matrix, for instance the computation of a kernel matrix for use in SVMs. The resultant matrix might suffer from data corruption as described in the Introduction. If this data corruption makes the matrix non-PSD, then it will probably render it unusable for the optimization algorithm. However, the matrix might still be close to another matrix that is PSD. In this section we review two ways of finding a convenient PSD approximation to the actual non-PSD matrix.

The *Frobenius norm* of a matrix $A_{n\times n}$ is defined as:

$$|| A ||_F = \sqrt{trace(A^T A)}.$$

The *Frobenius distance* between two compatible matrices is $d(A,B) = || A - B ||_F$. Equivalently, it can be computed as:

$$d^2(A,B) = \sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij} - b_{ij})^2$$

A necessary and sufficient condition for a matrix to be PSD is that its eigendecomposition has no negative eigenvalues. Spectral decomposition decomposes a matrix into a sum of matrices $\lambda_i u_i^T \times u_i$, where λ_i is the i-th eigenvalue, u_i the i-th eigenvector and $\times$ denotes outer product. To project a matrix into the PSD space, subtract from the original matrix those spectral matrices $u_i^T \times u_i$ with $\lambda_i < 0$. This is proven to be the closest approximation in Frobenius distance. This technique has been used in a wider algorithm called Alternating Projections or AP [4]. The AP algorithm offers some additional features, like the possibility of preserving the value of some elements of the matrix, like the diagonal, and still converge to a PSD matrix in a finite number of steps. For SVM this is irrelevant, but it may be of interest for other applications. We shall refer to this method as the *spectral decomposition* method.

Another way of forcing the PSD property on a matrix is to alter its values so as to increase the difference between the diagonal elements

and the non-diagonal elements. This is based on the following result:

A square matrix $A_{n\times n}$ is said to be *diagonally dominant* if

$$|a_{ii}| \geq \sum_{j\neq i}^{n} |a_{ij}|, 1 \leq i \leq n$$

It is known that symmetric and diagonally dominant matrices with nonnegative diagonal entries are PSD.

A very simple procedure to implement this idea is to progressively increase the diagonal elements until the matrix meets the PSD property. The disturbance of the matrix consists in multiplying the diagonal elements by some positive coefficient $1 < c$. The actual coefficient used is estimated as the smallest that renders a PSD matrix inside a certain threshold of accuracy[1]. We call this method the *matrix disturbance* method. We have chosen to increase the diagonal entries, though other applications may need to maintain a unitary diagonal; then, one could equivalently decrease the rest of the matrix multiplying off-diagonal entries by a coefficient $0 < c < 1$.

3 Missing values on PSD matrices

Given a kernel matrix that is PSD in nature and has some of its values missing, we wish to find a way to estimate the missing values in such a way that the completed matrix resembles as much as possible the original matrix. To do so, we discuss the nature of PSD matrices in order to find a general relation between the values of their entries.

3.1 Similarity Transitivity

Any symmetric PSD matrix K can be represented as a product of another matrix A and its transpose,

$$K = AA^T.$$

If we consider the matrix A as a collection of row vectors $(v_1 \dots v_n)$, then the value k_{ij} represents the product of vectors v_i and v_j. Hence, K can be seen as the correlation matrix of the system of vectors $(v_1 \dots v_n)$. If the vectors of A are normalized, that is, having a unitary norm, the product $\langle v_i, v_j \rangle$ is called the *cosine-similarity* of these vectors. This is the case when the diagonal elements k_{ii} of K are unitary. Cosine-similarity measures the angle between two unit vectors, so that the closer $\langle v_i, v_j \rangle$ is to 1 the more similar the vectors v_i and v_j are. When two vectors v_i and v_j have a high cosine-similarity (close to 1), their cosine-similarities to any other vector v_l will also be close. In symbols, for a unitary diagonal K, if $k_{ij} \approx 1$ then $k_{il} \approx k_{jl}$. This relation can be made precise as follows. Assume that $k_{ij} \geq 1 - \epsilon$ holds. Then,

$$\mid k_{il} - k_{jl} \mid = \mid \langle v_i, v_l \rangle - \langle v_j, v_l \rangle \mid =$$

$$\mid \langle v_i - v_j, v_l \rangle \mid \leq \mid\mid v_i - v_j \mid\mid \cdot \mid\mid v_l \mid\mid$$

where

$$\mid\mid v_i - v_j \mid\mid^2 = \mid\mid v_i \mid\mid^2 + \mid\mid v_j \mid\mid^2 - 2\langle v_i, v_j \rangle =$$

$$2 - 2k_{ij} \leq 2\epsilon.$$

so that

$$\mid k_{il} - k_{jl} \mid \leq \sqrt{2\epsilon}. \quad (1)$$

In the more general case, where K has non-unitary diagonal, the relationship is expressed as: if $s_{ij} \approx 1$ then $s_{il} \approx s_{jl}$ where s_{ij} is the general expression for the cosine-similarity of non-normalized vectors, so that we must divide by the norm of v_i and v_j. We know that the norm of the vector v_i is $\sqrt{k_{ii}}$, therefore

$$s_{ij} = \frac{k_{ij}}{\sqrt{k_{ii} \cdot k_{jj}}}$$

The plots in Figs. 1-4 show how this relationship holds for several example PSD matrices. These matrices are computed for several UCI benchmark data sets [5], as described in section 4. Each cross ('+') represents a pair v_i, v_j. The horizontal axis stands for k_{ij}. The vertical axis is the point-wise mean-square difference of the rows (i.e., the mean difference of the distances of each pair of vectors v_i, v_j to every other).

[1] The resulting matrix can be made PSD before reaching the diagonally dominant condition.

It can be seen how, as the value of k_{ij} increases, the mean-square difference of the rows decreases. In other words, the more similar v_i and v_j are, so is their relationship to the other vectors. This is an indication of data coherence. We call this relationship *similarity transitivity*.

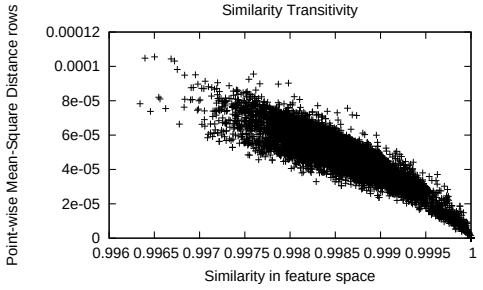

Figure 1: Similarity transitivity for Heart dataset.

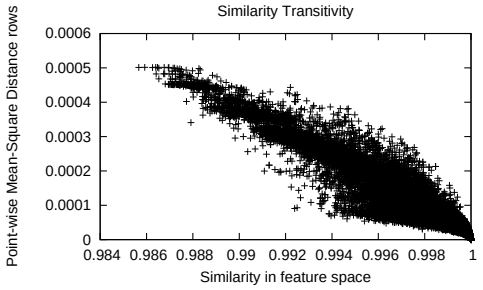

Figure 2: Similarity transitivity for Breast dataset.

3.2 Completing the missing values

The proposed strategy makes use of the principles described in Section 3.1 to estimate the missing elements of a matrix K, as follows:

Let us first consider the case where the PSD matrix K has unitary diagonal. Not only is this case easier to deal with, but it is often the case we come across in many applications (e.g., correlation matrices or normalized SVM kernel matrices). In this case, we propose the following formula to fill in a missing value k_{ij}.

First, the value of k_{ij} is partly built by looking at the values of k_{lj} for other vectors l.

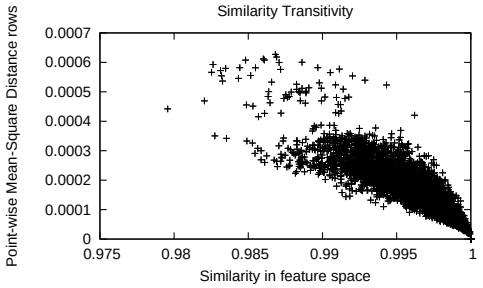

Figure 3: Similarity transitivity for Diabetes dataset.

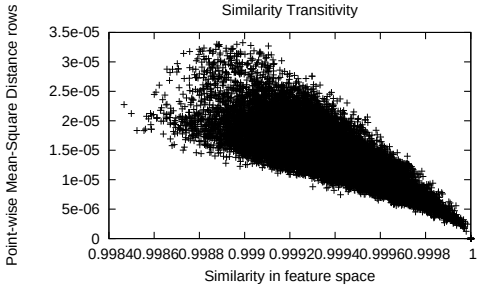

Figure 4: Similarity transitivity for German dataset.

Their influence is controlled by the value k_{il} (the cosine-similarity of v_i and v_l, as explained above). This amounts for the entire influence of example j. Conversely, and by symmetry, the influence of k_{li} is weighted by the similarity of the row vectors v_l and v_j of A, amounting for the entire influence of example i. Both terms are then averaged. The summation runs over those indices l such that the values k_{il} and k_{lj} are both available (not missing).

$$k'_{il} \equiv \frac{k_{il}}{\sum_l k_{il}}$$

$$k'_{jl} \equiv \frac{k_{jl}}{\sum_l k_{jl}}$$

$$k_{ij} \equiv \frac{1}{2} \sum_l k'_{il} k_{lj} + k'_{jl} k_{li}. \qquad (2)$$

It should be mentioned that nothing prevents the use of an averaging method other

than the arithmetic mean. In some contexts, a quadratic mean (squaring the averaged quantities and then taking the square root of their mean value), as well as the geometric and harmonic means could be used, if considered appropriate. In the same vein, the weighing factors k'_{il}, k'_{jl} can also play a significant role. For example, a quadratic function of the weights will make greater values of k'_{il}, k'_{jl} more prevalent. Those would be *soft* alternatives, whereas a *hard* one would be to use only the term with the highest weight. We don't consider this issue further in this work.

The previous estimate is devised for matrices with a unitary diagonal. In order to cope with matrices with non-unitary diagonal the analysis leads to a more general equation for the relation between entries:

$$s_{ij} \equiv \frac{1}{2} \sum_l \left(\frac{s_{il}}{\sum_l s_{il}} s_{lj} \right) + \left(\frac{s_{jl}}{\sum_l s_{jl}} s_{li} \right). \tag{3}$$

where

$$s_{ij} \equiv \frac{k_{ij}}{\sqrt{k_{ii} \cdot k_{jj}}},$$

An interesting problem then arises, for the values k_{ii} and k_{jj} could also be missing. For this reason, the diagonal must be first completed and then we can proceed with the rest. How to complete a diagonal element? For two very close vectors v_i, v_j in the cosine-similarity measure, the quotient of their dot product to another vector v_l would be close to the quotient of their norms, as the norms of the vector v_l cancel out. Formally, if $s_{ij} \approx 1$, then:

$$1 \approx \frac{s_{il}}{s_{jl}} = \frac{\frac{k_{il}}{\sqrt{k_{ii}}\sqrt{k_{ll}}}}{\frac{k_{jl}}{\sqrt{k_{jj}}\sqrt{k_{ll}}}} =$$

$$\frac{\frac{k_{il}}{\sqrt{k_{ii}}}}{\frac{k_{jl}}{\sqrt{k_{jj}}}} = \frac{k_{il}}{k_{jl}} \sqrt{\frac{k_{jj}}{k_{ii}}}$$

$$k_{ii} \approx (\frac{k_{il}}{k_{jl}})^2 k_{jj}$$

and therefore a missing k_{ii} can be estimated out of the non-missing k_{jj} as:

$$k_{ii} \equiv \sum_j \left(\frac{(k_{ij}/k_{jj})}{\sum_j (k_{ij}/k_{jj})} \frac{1}{N_l} \sum_l (\frac{k_{il}}{k_{jl}})^2 k_{jj} \right).$$

As before, the term (k_{ij}/k_{jj}) is used to weigh the sum, thus taking the more similar vectors more into account. The inner summation

$$\sum_l (\frac{k_{il}}{k_{jl}})^2 k_{jj}$$

runs for every l and given i, j granting that no value on the equation will be missing, and N_l represents the number of such terms considered. The outer summation runs only over values of j that also grant no missing terms.

Note that previously estimated values are used to estimate the new ones. It could be considered that new estimated values lack confidence enough to be used to estimate others. To compensate for this, the initially present values are exerting a constant influence throughout the process. Concerning special requisites for the matrices, notice that no negative values appear in this process if not already present on the original matrix. As to the values of the diagonal, if the matrix was of unitary diagonal nature, missing values in the diagonal are no problem since they are completed with a 1.

4 Experiments

The theoretical results concerning best PSD approximation in terms of the Frobenius distance leave open the question on how good is this estimation in practice. Further, we are interested in how the estimated or corrected matrices affect SVM performance.

In order to address these questions, we present two sets of experiments. In one set, covered in section 4.1, we are given a PSD matrix and artificially introduce several degrees of missingness. We aim at restoring the corrupted matrix using several strategies, and

then evaluate their performance. On the other set of experiments (section 4.2), corruption affects directly the input data, from which the kernel matrix is built out the of present values, thus rendering a non-PSD matrix. In this case, we also aim at coming up with a PSD matrix using several strategies and evaluate each in turn.

To evaluate how good a strategy is we make use of two metrics: the Frobenius distance between original and restored matrices and the difference in performance among these two matrices, as measured by cross-validation (CV) error. This last metric is particular to a SVM application. A near optimal model for a particular data-set and kernel is first found by means of grid exploration over the parameter space (regularization constant and kernel parameter), evaluating each possible combination with 5-fold CV. The 'optimal' combination is the one that delivers the best CV result. We thus consider the percentage of accuracy of this 'optimal' model as a measure of the discriminatory information contained in the data. When the matrix suffers from corruption part of this information is lost, therefore the subsequent 'optimal' model to be found will admittedly render a lower accuracy. The lower this accuracy, the more information has been lost.

All the data used in these experiments are drawn from the UCI database. The software used for the experiments is the libSVM package [6] for the SVM part. Data manipulation has been done using `perl` scripts binding with `Octave` [7], a GNU implementation of `Matlab`^{tm}. The 'optimal' model for a dataset has been computed using a grid exploration script provided with the libSVM package. The used kernel is the Radial Basis Function (RBF) kernel:

$$RBF(x,y) = \exp^{-\gamma\,||x-y||^2}, \qquad \gamma > 0$$

4.1 Missing values on kernel matrices

These experiments are composed of several runs. In each of the runs we take an original PSD matrix and replace a number of random locations with missing values. We then take the incomplete matrix and estimate the missing values following the similarity transitivity (ST) strategy or simple mean imputation (MI). The final step is to force the PSD property again, which is done by spectral decomposition or by matrix disturbance. The resultant matrix is then compared to the original using the Frobenius distance, and then its information is measured using 5-fold CV.

The original matrix is computed using the RBF kernel and the γ parameter estimated for the 'optimal' model using grid exploration. The quantity of missing values inserted into the original matrix is a number from 0% to 75% increasing in 5% steps. To find the model for each restored matrix, and thus to measure its information, we explore a range of capacities around the 'optimal' capacity for the original matrix. The values used are $0.10, 0.5, 2, 1, 5, 10$ or 100 times the original 'optimal' capacity.

The experiments above have been run for four different widely-used UCI databases (*Breast Cancer, German Credit, Pima Indians Diabetes, Heart Disease*). We present the following plots, shown in one column for each dataset, for clarity:

- *Cross-Validation.* The accuracy of the 'optimal' model for 5-fold Cross Validation (Figs. 5, 8, 11 & 14).
- *Frobenius using ST estimate.* Frobenius distance between the original matrix and three other matrices: (i) the matrix after completing the missing values using ST *before* restoring the PSD property, (ii) the same matrix *after* restoring it by spectral decomposition and (iii) the same one restored using matrix disturbance. (Figs. 6, 9, 12 & 15).
- *Frobenius using MI estimate.* Frobenius distance between the original matrix and the matrix completed using MI and restoring using spectral decomposition. These plots are separated from rest for readability, as the values are much higher. (Figs. 7, 10, 13 & 16).

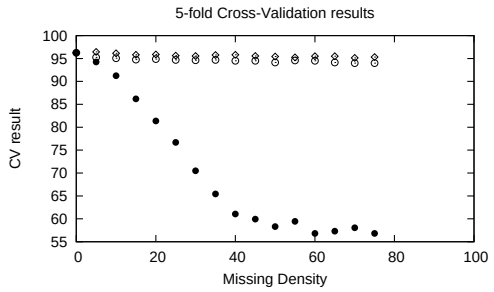

Figure 5: **Breast Cancer Cross-Validation results:** (○) ST estimates plus disturbance restoring, (◇) ST estimate and spectral restoring, (•) mean estimates and spectral restoring.

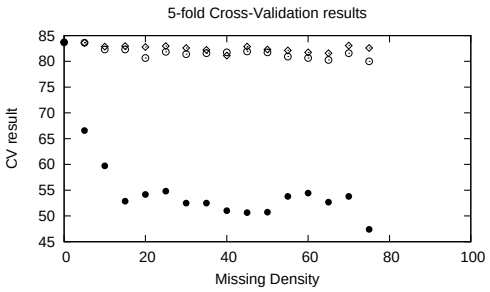

Figure 8: **Heart Disease Cross-Validation results:** (○) ST estimates plus disturbance restoring, (◇) ST estimate and spectral restoring, (•) mean estimates and spectral restoring.

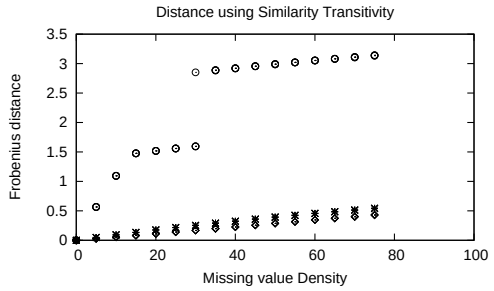

Figure 6: **Breast Cancer Frobenius distance results:** (◇) ST estimates and spectral restoring, (○) ST estimates plus disturbance restoring, (∗) ST estimate with no PSD restoring.

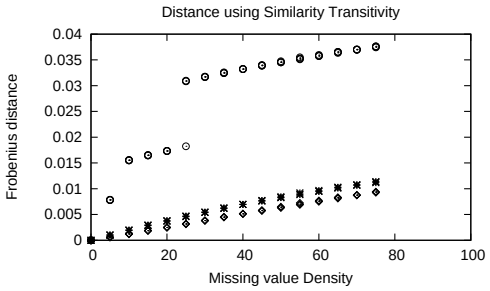

Figure 9: **Heart Disease Frobenius distance results:** (◇) ST estimates and spectral restoring, (○) ST estimates plus disturbance restoring, (∗) ST estimate with no PSD restoring.

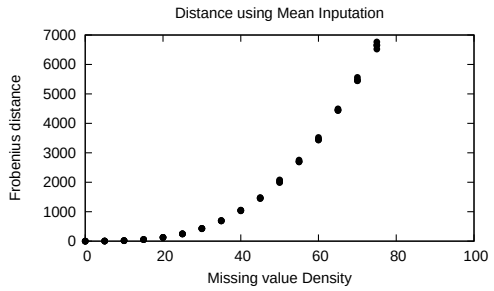

Figure 7: **Breast Cancer Frobenius distance results:** The plot (•) represents mean estimates with spectral restoring.

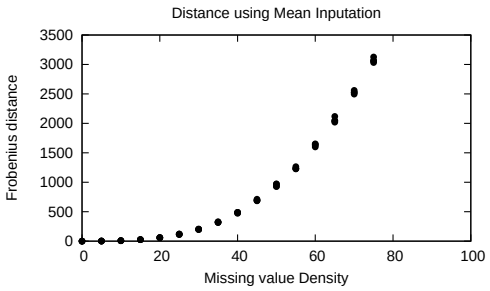

Figure 10: **Heart Disease Frobenius distance results:** The plot (•) represents mean estimates with spectral restoring

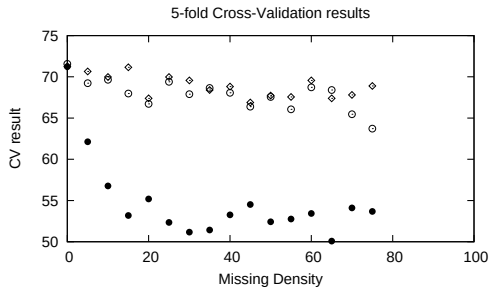

Figure 11: **Pima Diabetes Cross-Validation results:** (○) ST estimates plus disturbance restoring, (◇) ST estimate and spectral restoring, (•) mean estimates and spectral restoring.

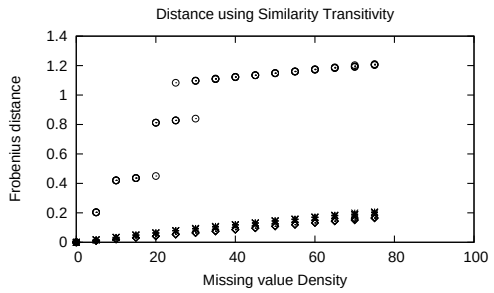

Figure 12: **Pima Diabetes Frobenius distance results:** (◇) ST estimates and spectral restoring, (○) ST estimates plus disturbance restoring, (∗) ST estimate with no PSD restoring.

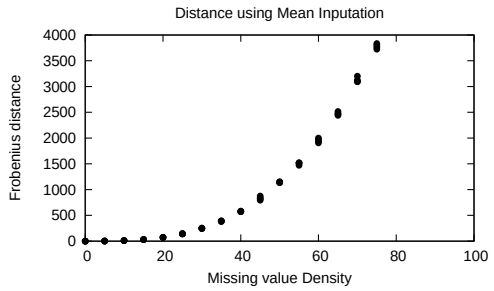

Figure 13: **Pima Diabetes Frobenius distance results:** The plot (•) represents mean estimates with spectral restoring

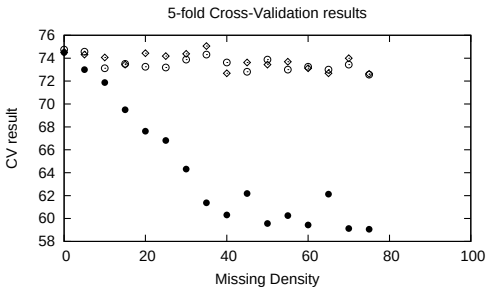

Figure 14: **German Credit Cross-Validation results:** (○) ST estimates plus disturbance restoring, (◇) ST estimate and spectral restoring, (•) mean estimates and spectral restoring.

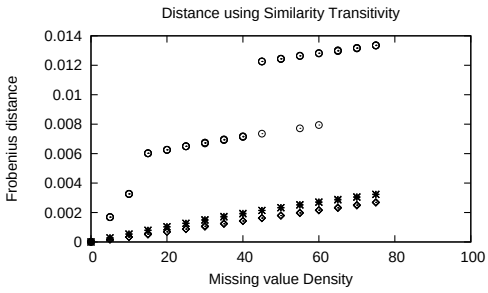

Figure 15: **German Credit Frobenius distance results:** (◇) ST estimates and spectral restoring, (○) ST estimates plus disturbance restoring, (∗) ST estimate with no PSD restoring.

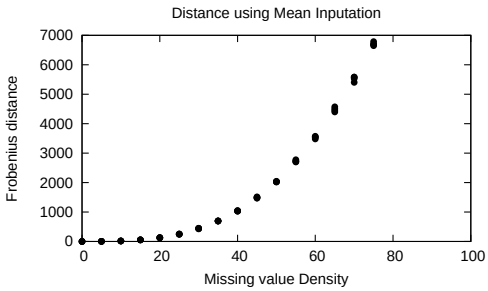

Figure 16: **German Credit Frobenius distance results:** The plot (•) represents mean estimates with spectral restoring

The results for the Frobenius distance show clearly that the ST approach to missing value completion outperforms the mean imputation immensely. The spectral decomposition strategy also performs better than the matrix disturbance. In fact, an interesting thing is noticeable: the Frobenius distance of the completed matrix decreases after the spectral restoration, although only slightly, which means that this step not only provides the missing PSD property but also gets us closer to the original matrix.

The 'optimal' Cross Validation model accuracy metric also shows a clear advantage for the similarity transitivity technique compared to the mean imputation. However it shows no clear advantage for spectral decomposition over matrix disturbance.

4.2 Missing values on input data

The second set of experiments conducted deal with the scenario of corruption over the input data fed to the SVM algorithm. In this particular situation there are several basic alternatives:

- *List-wise data deletion:* Removal of all examples with missing data. We do not consider this alternative at all, as it can lead to excessive information loss.
- *Pairwise data deletion:* For each pair of input vectors for which the kernel is evaluated, consider only the present values in both of them. The result will be a non PSD matrix. Spectral decomposition can then be used to render the matrix usable.
- *Mean imputation:* Fill in each missing value with the mean of the whole row or column. It might not be the best approximation, but it allows to work with the data right away.

We consider three different approaches for this problem: (i) use mean imputation, compute the kernel from the resultant data and evaluate it, (ii) use pairwise deletion plus spectral decomposition to restore the PSD matrix; and (iii) add a step between pairwise deletion and spectral decomposition by readjusting the values that have suffered the highest pairwise deletion. This readjustment is done based on other values of the matrix that have been less affected by deletion and are close to the one in hand, using the similarity transitivity principle. The final value is the mean of the initial value and of those that have more confidence around it. The mean is weighed by the amount of pairwise deletion and closeness to the value.

Rather susprisingly, the results indicate that for the Frobenius distance the MI method shows slightly better results than the other more complex alternatives. Also the extra readjustment step show little or no improvement over the standard pairwise alternative. As for the final accuracy, all alternatives show very similar results. No plots have been included on this experiment for space reasons, as we consider them inconclusive and of little interest.

5 Conclusions

This article shows how simple properties of Positive Semi-Definite (PSD) matrices can be used to deal with missing values. We have reviewed two methods to force the PSD property and a method to fill in missing values explicitly designed for PSD matrices, which has been named Similarity Transitivity. We have described several situations and investigated their applicability.

In order to illustrate the methods, a set of experiments involving support vector machine (SVM) classification problems has been presented and discussed. The performance of the methods has been evaluated using Frobenius (point-to-point) distance to the original matrix and mean Cross-Validation error.

Similarity Transitivity seems like a strong relationship that holds between values of a PSD matrix. Its use in filling in missing values for PSD matrices has shown to yield very good results. However this is a fairly uncommon case in SVM; in a normal training scenario we are given the input data, not the kernel matrix. As for other scenarios, missing values in input data, no performance increase was

achieved.

We understand the PSD matrices are fairly common in a number of scientific situations, so the results presented here can hopefully be of help for applications other than SVMs.

6 Appendix: pseudocode

- Pseudocode for missing value estimation in unitary diagonal matrices.

```
for i in rows do
  for j in columns do
    if K[i,j] is missing then
       new_val <- 0
       sum <- 0
       for l in rows do
         if K[l,j] is not missing then
           new_val <- new_val + K[i,l]*K[l,j]
           sum <- sum + K[i,l]
         endif
       endfor
       new_val <- new_val/sum
       K[i,j] <- new_val
       K[j,i] <- new_val
    endif
  endfor
endfor
```

- Pseudocode for off-diagonal missing value estimation in matrices with non-unitary diagonal (sq denotes square root).

```
for i in rows do
  for j in columns do
    if K[i,j] is missing then
      new_val <- 0
      sum <- 0
      for l in rows do
       if K[l,j] is not missing then
         new_val <- new_val +
          K[i,l]/sq(K[i,i]*K[l,l])*
          K[l,j]/sq(K[l,l]*K[j,j])
         sum <- sum+K[i,l]/sq(K[i,i]*K[l,l])
       endif
      endfor
      new_val <- new_val/(sum*sq(K_ii*K_jj))
      K[i,j] <- new_val
      K[j,i] <- new_val
    endif
  endfor
endfor
```

- Pseudocode for diagonal missing value estimation in matrices with non-unitary diagonal.

```
for i in rows do
 if K[i,i] is missing then
   sum <- 0;
   acc <- 0;
   for j in rows do
    if K[j,i],K[j,j] are not missing then
      for l in rows do
       if K[j,l],K[i,l] are not missing then
         sum <- sum + K[i,j]
         acc <- K[i,j]*K[j,j]*K[i,l]/K[j,l]
       endif
      endfor
    endif
   endfor
   K[i,i] <- acc/sum;
 endif
endfor
```

References

[1] C.Cortes and V.Vapnik. Support vector networks. *Machine Learning* (20), 1995.

[2] V.Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[3] D. L. Medin, R. L. Goldstone, and D. Gentner. Respects for similarity. *Psychological Review*, 100(2):254 278, 1993.

[4] C. Lucas. *Computing nearest covariance and correlation matrices*. Ph.D. Thesis, Univ. of Manchester, 2001.

[5] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, Dept. of Information and Computer Science, University of California, Irvine, CA, 1998.

[6] LIBSVM: A Library for Support Vector Machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[7] `http://www.octave.org`